Numerical methods for conservation laws. Application to gas dynamics and plasma physics

Emmanuel Franck

Soutenance d'habilitation à diriger des recherches
IRMA Strasbourg, 17 Janvier 2023

[1]Inria Nancy Grand Est, France
[2]IRMA, Strasbourg University, France

Outline

Introduction

Central problem: stiff hyperbolic and kinetic equations

General hyperbolic system

- We consider $\boldsymbol{U}(t,x) \in \mathbb{R}^n$. General equations:

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{S}(\boldsymbol{U})$$

with $\boldsymbol{F}(\boldsymbol{U})$ the flux and $\boldsymbol{S}(\boldsymbol{U})$ the source term.

Central problem: stiff hyperbolic and kinetic equations

General hyperbolic system

- We consider $\boldsymbol{U}(t,x) \in \mathbb{R}^n$. General equations:

$$\partial_t \boldsymbol{U} + \frac{1}{\epsilon}\partial_x \boldsymbol{F}(\boldsymbol{U}) = \frac{1}{\epsilon}\boldsymbol{S}(\boldsymbol{U})$$

- **Limit**: long time limit
- **Scaling**: $\tilde{t} = \epsilon t$,

Limit

- Asymptotic limit:

$$\partial_x \mathbf{F}(\mathbf{U}) = \mathbf{S}(\mathbf{U})$$

- **Exemple**: Shallow Water equations with topography and vanishing initial velocity.

$$\left\{ \begin{array}{l} \partial_t h + \partial_x(hu) = 0 \\ \partial_t hu + \partial_x(hu^2 + gh^2) = h\partial_x z \end{array} \right. \rightarrow \left\{ \begin{array}{l} u = 0 \\ h\partial_x(h+z) = 0 \end{array} \right.$$

Central problem: stiff hyperbolic and kinetic equations

General hyperbolic system

- We consider $\boldsymbol{U}(t,x) \in \mathbb{R}^n$. General equations:

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \frac{\sigma}{\epsilon} \boldsymbol{R}(\boldsymbol{U})$$

with $\boldsymbol{S}(\mathbf{U}) = \sigma \boldsymbol{R}(\boldsymbol{U})$

- **Limit**: Relaxation limit,
- **Scaling**: $\tilde{\sigma} = \epsilon\sigma$

Limit

- Asymptotic limit:

$$\partial_t \mathbf{u} + \partial_x \mathbf{G}(\mathbf{u}) = 0$$

with $\mathbf{u} \in \mathbb{R}^d$ wiht $d < n$.

- **Exemple**: Vlasov-Poisson in the collisional regime.

$$\left\{ \begin{array}{l} \partial_t f + v\partial_x f + \partial_x \phi \partial_v f = \frac{\sigma}{\epsilon}(M_{\rho,u,T} - f) \\ -\Delta\phi = \int f \end{array} \right. \rightarrow \left\{ \begin{array}{l} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t \rho u + \partial_x(\rho u^2 + p) = 0 \\ \partial_t E + \partial_x(Eu + pu) = 0 \end{array} \right.$$

Central problem: stiff hyperbolic and kinetic equations

General hyperbolic system

- We consider $\boldsymbol{U}(t,x) \in \mathbb{R}^n$. General equations:

$$\partial_t \boldsymbol{U} + \frac{1}{\epsilon}\partial_x \boldsymbol{F}(\boldsymbol{U}) = \frac{\sigma}{\epsilon^2}\boldsymbol{R}(\boldsymbol{U})$$

with $\boldsymbol{S}(\boldsymbol{U}) = \sigma \boldsymbol{R}(\boldsymbol{U})$

- **Limit**: Diffusion limit,
- **Scaling**: $\tilde{\sigma} = \epsilon\sigma, \tilde{t} = \epsilon t$

Limit

- Asymptotic limit:

$$\partial_t \mathbf{u} + \partial_x \mathbf{G}(\mathbf{u}) = \partial_x(\mathbf{D}(\mathbf{u})\partial_x \mathbf{u})$$

with $\mathbf{u} \in \mathbb{R}^d$ wiht $d < n$.

- **Exemple**: P_1 radiative model.

$$\left\{ \begin{array}{l} \partial_t E + \frac{1}{\epsilon}\partial_x F = 0 \\ \partial_t F + \frac{1}{3\epsilon}\partial_x E = -\frac{\sigma}{\epsilon^2} F \end{array} \right. \rightarrow \partial_t E - \partial_x \left(\frac{1}{3\sigma}\partial_x E \right) = 0$$

Central problem: stiff hyperbolic and kinetic equations

General hyperbolic system

- We consider $\boldsymbol{U}(t,x) \in \mathbb{R}^n$. General equations:

$$\partial_t \boldsymbol{U} + \frac{1}{\epsilon}\partial_x \mathbf{F}_f(\boldsymbol{U}) + \partial_x \mathbf{F}_l(\boldsymbol{U}) = \boldsymbol{S}(\boldsymbol{U})$$

with $\boldsymbol{F}(\boldsymbol{U}) = \frac{1}{\epsilon}\mathbf{F}_f(\boldsymbol{U}) + \mathbf{F}_l(\boldsymbol{U})$.

- **Limit**: fast wave limit,
- **Scaling**: $\tilde{\lambda} = \epsilon\lambda$ with λ the maximal eigenvalue of $\mathbf{F}(\mathbf{U})$.

Limit

- Asymptotic limit (very formal):

$$\begin{cases} \partial_t \boldsymbol{U} + P\partial_x \boldsymbol{F}_l(\boldsymbol{U}) = \boldsymbol{S}(\boldsymbol{U}) \\ \partial_x \boldsymbol{F}_f(\mathbf{U}) = 0 \end{cases}$$

with P the projector on $\ker(\partial_x \mathbf{F}_f(\mathbf{U}))$

- **Example**: Euler in the low Mach limit (next section).
- [*Gui*15] for reduced MHD.

Numerical difficulties associated to stiff problems

Spatial discretization

- For hyperbolic systems, we use classical Finite Volume schemes.
- For equations such as

$$\partial_t \boldsymbol{U} + \frac{1}{\epsilon^\alpha}\partial_x \boldsymbol{F}(\boldsymbol{U}) = \frac{1}{\epsilon^\beta}\boldsymbol{S}(\boldsymbol{U}),$$

the equivalent equations associated with FV schemes are of the form:

$$\partial_t \boldsymbol{U} + \frac{1}{\epsilon^\alpha}\partial_x \boldsymbol{F}(\boldsymbol{U}) = \frac{\Delta x}{\epsilon^\alpha}\partial_x(A(\boldsymbol{U})\partial_x \boldsymbol{U}) + \frac{1}{\epsilon^\beta}\boldsymbol{S}(\boldsymbol{U}) + O(\Delta x^2),$$

with A the viscosity matrix.

- In general, the error is in $O\left(\frac{\Delta x}{\epsilon^\alpha}\right)$.

Time scheme

In general, for stiff hyperbolic systems, we use explicit schemes. They come with a CFL condition like:

$$\Delta t < \min(\epsilon^\alpha \Delta x, \epsilon^\beta)$$

Main issue

Since Δx or/and Δt scale with ϵ, at fixed accuracy the CPU grow up with $\frac{1}{\epsilon}$ or $\frac{1}{\epsilon^2}$.

Implicit relaxation schemes for PDEs

Euler equations and the low Mach regime

- **Euler equations**:

$$\begin{cases} \partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0 \\ \partial_t (\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u} + p I_d) = 0 \\ \partial_t E + \nabla \cdot (E \boldsymbol{u} + p \boldsymbol{u}) = 0 \end{cases}$$

  with $\rho(t, \mathbf{x}) > 0$ the density, $\boldsymbol{u}(t, \mathbf{x})$ the velocity and $E(t, \mathbf{x}) > 0$ the total energy.

- **Hyperbolic system** with nonlinear waves. **Waves speed**: three differents eigenvalues: $(\boldsymbol{u}, \mathbf{n})$ and $(\boldsymbol{u}, \mathbf{n}) \pm c$ with the sound speed $c^2 = \gamma \frac{p}{\rho}$.

Euler equations and the low Mach regime

- **Euler equations**:

$$\begin{cases} \partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0 \\ \partial_t (\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u} + p I_d) = 0 \\ \partial_t E + \nabla \cdot (E \boldsymbol{u} + p \boldsymbol{u}) = 0 \end{cases} \longrightarrow \begin{cases} \partial_t \rho + \nabla \cdot (\rho \boldsymbol{u}) = 0 \\ \partial_t (\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \frac{1}{M^2} \nabla p = 0 \\ \partial_t E + \nabla \cdot (E \boldsymbol{u} + p \boldsymbol{u}) = 0 \end{cases}$$

  with $\rho(t, \mathbf{x}) > 0$ the density, $\boldsymbol{u}(t, \mathbf{x})$ the velocity and $E(t, \mathbf{x}) > 0$ the total energy.

- **Hyperbolic system** with nonlinear waves. **Waves speed**: three differents eigenvalues: $(\boldsymbol{u}, \mathbf{n})$ and $(\boldsymbol{u}, \mathbf{n}) \pm c$ with the sound speed $c^2 = \gamma \frac{p}{\rho}$.

Physic interpretation:

- **Two important velocity scales**: $\boldsymbol{u}$ and c, and their ratio (the Mach number) $M = \frac{|\boldsymbol{u}|}{c}$.
- When M tends to zero, we obtain the incompressible Euler equations:

$$\begin{cases} \partial_t \rho + \boldsymbol{u} \cdot \nabla \rho = 0 \\ \rho \partial_t \boldsymbol{u} + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p_2 = 0 \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases}$$

  In 1D, we only have an advection of ρ.
- **Aim**: construct an scheme valid at the limit with a uniform cost compared to M.
- **Other related problems**: Euler with gravity, low-Mach and low-β MHD.

Explicit vs implicit schemes

- **Explicit scheme**: issues with the CFL condition for low Mach flow:
  - Fast perturbative phenomena: acoustic waves at velocity c
  - Important phenomena: transport at velocity u
  - Expected CFL condition $\Delta t < \frac{\Delta x}{|u|}$; in practice, we need $\Delta t < \frac{\Delta x}{c} = M\frac{\Delta x}{|u|}$
  - At the end, we need a Δt multiplied by M compared to the expected Δt

First solution

Implicit time scheme. No CFL condition. Taking a larger time step, it allows to "filter" the fast acoustic waves which are not import to capture the limit regime.

- Implicit scheme: Newton method + GMRES
- Simpler example (linearized Euler equations around $\mathbf{u}_0 = 0$):

$$\left\{ \begin{array}{l} \partial_t p + \frac{1}{M}\nabla\cdot\mathbf{u} = 0 \\ \partial_t \mathbf{u} + \frac{1}{M}\nabla p = 0 \end{array} \right. \rightarrow \left\{ \begin{array}{l} p^{n+1} + \frac{\Delta t}{M}\nabla\cdot\mathbf{u}^{n+1} = p^n \\ \mathbf{u}^{n+1} + \frac{\Delta t}{M}\nabla p^{n+1} = \mathbf{u}^n \end{array} \right.$$

- Matrix to invert:

$$\begin{pmatrix} \frac{M}{\Delta t} I_d & \nabla\cdot \\ \nabla & \frac{M}{\Delta t} I_d \end{pmatrix}$$

- If $\Delta t \gg M$, the problem is ill-posed, and the matrix is difficult to invert.

First idea: physics-based preconditioning

- Coming back to the example, consider the parabolization by Knoll-Keyes [04] and Chacon [08-10]:

$$\begin{cases} p^{n+1} + \dfrac{\Delta t}{M} \nabla \cdot \mathbf{u}^{n+1} = p^n \\ \mathbf{u}^{n+1} + \dfrac{\Delta t}{M} \nabla p^{n+1} = \mathbf{u}^n \end{cases} \rightarrow \begin{cases} p^{n+1} - \dfrac{\Delta t^2}{M^2} \Delta p^{n+1} = \ldots \\ \mathbf{u}^{n+1} = \mathbf{u}^n - \dfrac{\Delta t}{M} \nabla p^{n+1} \end{cases}$$

- The matrix to invert becomes:

$$\left(\frac{M^2}{\Delta t^2} I_d - \Delta \right)$$

- If $\Delta t \gg M$, we invert a Laplacian (for instance with multigrid (MG) approaches).

Physics-based preconditioning

Use the same parabolization approach with approximation for nonlinear problems like Euler or MHD, solve the parabolic model with MultiGrid and use this solving as right preconditioning.

Remarks

The approach is efficient but boundary conditions are tricky, and analyzing failures is difficult.

Second idea: relaxation approach

Relaxation approach

Keep the idea to replace the original model by one that is simpler to solve, use it as a solver rather than a preconditioner.

- **Relaxation** [JX95] : Used to design new schemes.
- **Idea**: Approximate the model

$$\partial_t \boldsymbol{U} + \partial_x \mathbf{F}(\boldsymbol{U}) = 0, \qquad \text{by} \qquad \partial_t \mathbf{f} + \mathbf{A}(\mathbf{f}) = \frac{1}{\varepsilon}(Q(\mathbf{f}) - \mathbf{f})$$

At the limit (Hilbert expansion) and taking $P\mathbf{f} = \boldsymbol{U}$ ($P \in \mathbb{R}^{n,m}$ with $n < m$) we obtain

$$\partial_t \boldsymbol{U} + \partial_x \mathbf{F}(\boldsymbol{U}) = \varepsilon \partial_x (D(\boldsymbol{U}) \partial_x \boldsymbol{U}) + O(\varepsilon^2)$$

- **Time scheme**: Splitting
  - We first solve

$$\frac{\mathbf{f}^* - \mathbf{f}^n}{\Delta t} + \mathbf{A}(\mathbf{f}^{*,n}) = 0,$$

  - We solve the stiff source term using an implicit scheme.

Advantages of this approach

- In general, we construct $\mathbf{A}$ with a simpler structure than $\mathbf{F}$, to easily designed a Godunov numerical flux.
- Here we use it to construct some simpler implicit schemes.

Xin-Jin relaxation method

- We consider the following nonlinear hyperbolic system

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = 0,$$

with a function $\boldsymbol{U} \in \mathbb{R}^N$, $x \in \mathbb{R}^d$.

- **Aim**: Find a way to approximate this system with a sequence of simple systems.
- **Idea**: Xin-Jin relaxation method (very popular in the hyperbolic and Finite Volume community) [JX95]-[Nat96]-[ADN00].

$$\begin{cases} \partial_t \boldsymbol{U} + \partial_x \boldsymbol{V} = 0 \\ \partial_t \boldsymbol{V} + \lambda^2 \partial_x \boldsymbol{U} = \dfrac{1}{\varepsilon}(\boldsymbol{F}(\boldsymbol{U}) - \boldsymbol{V}) \end{cases}$$

Limit scheme for the hyperbolic relaxation

The limit equation of the relaxation system is

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \varepsilon \partial_x((\lambda^2 Id - |A(\boldsymbol{U})|^2)\partial_x \boldsymbol{U}) + O(\varepsilon^2),$$

with $A(\boldsymbol{U})$ the Jacobian of $\boldsymbol{F}(\boldsymbol{U})$.

- **Conclusion**: the relaxation system is an approximation of the original hyperbolic system (with an error in ε).

Xin-Jin implicit scheme

Main property

- **Relaxation system**: "the nonlinearity is local and the non-locality is linear".
- **Main idea**: **splitting scheme** between implicit transport and **implicit** relaxation.
- **Key point**: we have $\partial_t \boldsymbol{U} = 0$ during the relaxation step. Therefore $\boldsymbol{F}(\boldsymbol{U})$ **is explicit**.

- Relaxation step: we use a θ scheme:

$$\begin{cases} \boldsymbol{U}^{n+1} = \boldsymbol{U}^n \\ \boldsymbol{V}^{n+1} = \theta \dfrac{\Delta t}{\varepsilon}(\boldsymbol{F}(\boldsymbol{U}^{n+1}) - \boldsymbol{V}^{n+1}) + (1-\theta)\dfrac{\Delta t}{\varepsilon}(\boldsymbol{F}(\boldsymbol{U}^n) - \boldsymbol{V}^n) \end{cases}$$

Xin-Jin implicit scheme

Main property

- **Relaxation system**: "the nonlinearity is local and the non-locality is linear".
- **Main idea**: **splitting scheme** between implicit transport and **implicit** relaxation.
- **Key point**: we have $\partial_t \boldsymbol{U} = 0$ during the relaxation step. Therefore $\boldsymbol{F}(\boldsymbol{U})$ **is explicit**.

- Relaxation step: we use a θ scheme:

$$\begin{cases} \boldsymbol{U}^{n+1} = \boldsymbol{U}^n \\ \boldsymbol{V}^{n+1} = \theta \dfrac{\Delta t}{\varepsilon}(\boldsymbol{F}(\boldsymbol{U}^n) - \boldsymbol{V}^{n+1}) + (1-\theta)\dfrac{\Delta t}{\varepsilon}(\boldsymbol{F}(\boldsymbol{U}^n) - \boldsymbol{V}^n) \end{cases}$$

Xin-Jin implicit scheme

Main property

- **Relaxation system**: "the nonlinearity is local and the non-locality is linear".
- **Main idea**: **splitting scheme** between implicit transport and **implicit** relaxation.
- **Key point**: we have $\partial_t \boldsymbol{U} = 0$ during the relaxation step. Therefore $\boldsymbol{F}(\boldsymbol{U})$ **is explicit**.

- Relaxation step: we use a θ scheme:

$$\begin{cases} \boldsymbol{U}^{n+1} = \boldsymbol{U}^n \\ \left(I_d + \theta \dfrac{\Delta t}{\varepsilon} \right) \boldsymbol{V}^{n+1} = \theta \dfrac{\Delta t}{\varepsilon} \boldsymbol{F}(\boldsymbol{U}^n) + (1-\theta) \dfrac{\Delta t}{\varepsilon} (\boldsymbol{F}(\boldsymbol{U}^n) - \boldsymbol{V}^n) \end{cases}$$

Xin-Jin implicit scheme

Main property

- **Relaxation system**: "the nonlinearity is local and the non-locality is linear".
- **Main idea**: **splitting scheme** between implicit transport and **implicit** relaxation.
- **Key point**: we have $\partial_t \boldsymbol{U} = 0$ during the relaxation step. Therefore $\boldsymbol{F}(\boldsymbol{U})$ **is explicit**.

- Relaxation step: we use a θ scheme:

$$\begin{cases} \boldsymbol{U}^{n+1} = \boldsymbol{U}^n \\ \boldsymbol{V}^{n+1} = \boldsymbol{V}^n + \underbrace{\dfrac{\Delta t}{\varepsilon + \theta \Delta t}}_{\omega} (\boldsymbol{F}(\boldsymbol{U}^n) - \boldsymbol{V}^n) \end{cases}$$

Xin-Jin implicit scheme

Main property

- **Relaxation system**: "the nonlinearity is local and the non-locality is linear".
- **Main idea**: **splitting scheme** between implicit transport and **implicit** relaxation.
- **Key point**: we have $\partial_t \boldsymbol{U} = 0$ during the relaxation step. Therefore $\boldsymbol{F}(\boldsymbol{U})$ **is explicit**.

- Relaxation step: we use a θ scheme:
- Transport step (order 1) :

$$\left(I_d + \Delta t \begin{pmatrix} 0 & 1 \\ \lambda^2 & 0 \end{pmatrix} \partial_x \right) \begin{pmatrix} \boldsymbol{U}^{n+1} \\ \boldsymbol{V}^{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}^{n} \\ \boldsymbol{V}^{n} \end{pmatrix}$$

We plug the equation on $\boldsymbol{V}$ in the equation on $\boldsymbol{U}$ and obtain

$$(I_d - \Delta t^2 \lambda^2 \partial_{xx}) \boldsymbol{U}^{n+1} = \boldsymbol{U}^n - \Delta t \partial_x \boldsymbol{V}^n, \quad \boldsymbol{V}^{n+1} = \boldsymbol{V}^n - \Delta t \lambda^2 \partial_x \boldsymbol{U}^{n+1}$$

Numerical error of first splitting scheme

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \Delta t \left(\frac{2-\omega}{\omega} \right) \partial_x ((\lambda^2 I_d - |A(\boldsymbol{U})|^2) \partial_x \boldsymbol{U}) + O(\Delta t^2)$$

Generic kinetic relaxation schemes

Kinetic relaxation systems

- **Model under consideration**:
$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = 0$$
- **Lattice**: $W = \{\lambda_1, ..., \lambda_{n_v}\}$ a set of velocities.
- **Mapping matrix**: P a matrix $n_c \times n_v$ $(n_c < n_v)$ such that $\boldsymbol{U} = P\boldsymbol{f}$, with $\boldsymbol{U} \in \mathbb{R}^{n_c}$.
- **Kinetic relaxation system**:
$$\partial_t \boldsymbol{f} + \Lambda \partial_x \boldsymbol{f} = \frac{1}{\varepsilon}(\boldsymbol{f}^{eq}(\boldsymbol{U}) - \boldsymbol{f})$$
- Consistency condition (Natalini - Aregba [96-98-02], Bouchut [99-03]) :
$$\left\{ \begin{array}{lcl} P\boldsymbol{f}^{eq}(\boldsymbol{U}) & = & \boldsymbol{U} \\ P\Lambda\boldsymbol{f}^{eq}(\boldsymbol{U}) & = & \boldsymbol{F}(\boldsymbol{U}) \end{array} \right. \qquad (\mathcal{C})$$

Chapman-Enskog stability

- **Limit system**:
$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \varepsilon \partial_x \left(\left(P\Lambda^2 \partial_{\boldsymbol{U}} \boldsymbol{f}^{eq}(\boldsymbol{U}) - |\partial \boldsymbol{F}(\boldsymbol{U})|^2 \right) \partial_x \boldsymbol{U} \right) + O(\varepsilon^2)$$
- This limit system is stable if the second order operator is entropy-dissipative. We also have partial stability results for the kinetic systems.
- **Strong Stability**: entropy theory equivalent to the H-theorem. Other criteria for stability are given in Bouchut [04].

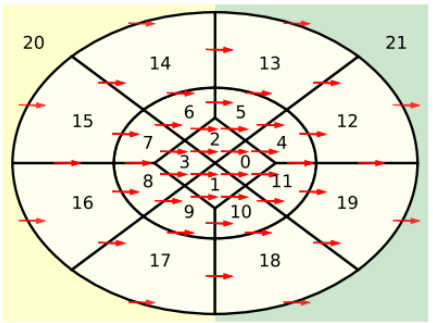
Implicit scheme based on kinetic relaxation I

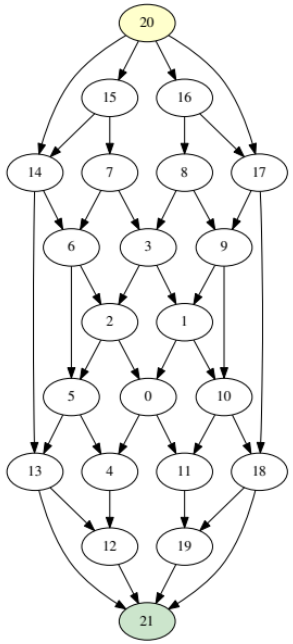
- We define the two operators for each step :

$$T(\Delta t) : e^{\Delta t \Lambda \partial_x} \boldsymbol{f}^{n+1} = \boldsymbol{f}^n$$

$$R(\Delta t) : \boldsymbol{f}^{n+1} = \boldsymbol{f}^n + \omega(\boldsymbol{f}^{eq}(\boldsymbol{U}^n) - \boldsymbol{f}^n)$$

- **First splitting scheme**: $T(\Delta t) \circ R(\Delta t)$ is consistent with
- How to deal with the transport step with constant velocity?
  - Semi-Lagrangian scheme,
  - CFL-less implicit DG scheme, with a downwind strategy: block triangular matrix using task graph numbering.





- **Boundary conditions**: studied in Drui and al [21].

Implicit scheme based on kinetic relaxation II

High order scheme: composition method

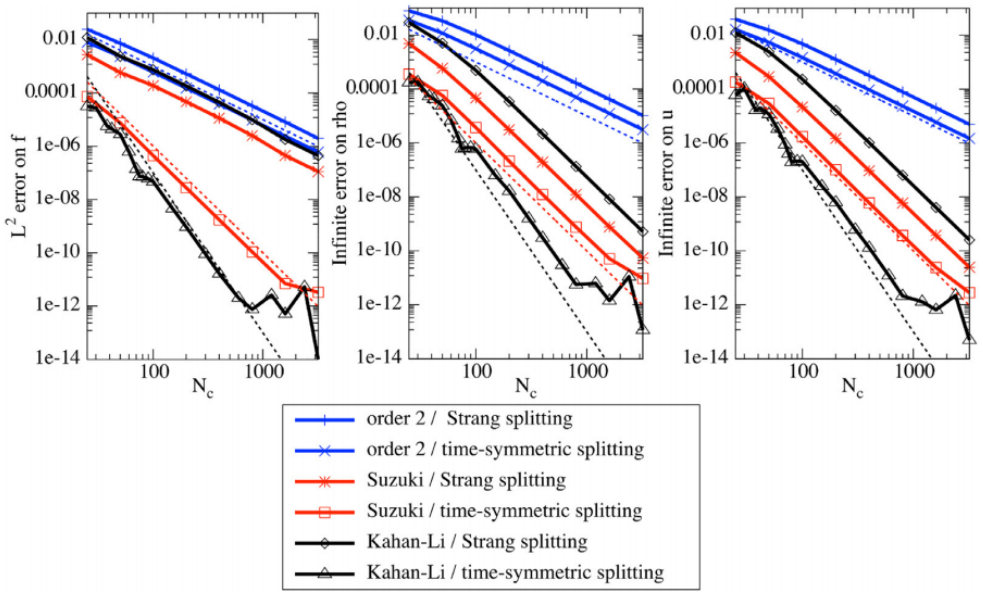
- If Ψ, a scheme that is second-order accurate in time, satisfies $\Psi(\Delta t) = \Psi^{-1}(-\Delta t)$ and $\Psi(0) = I_d$, then we can construct the high-order extension
$$M_p(\Delta t) = \Psi(\gamma_1 \Delta t) \circ \Psi(\gamma_2 \Delta t) \circ \cdots \circ \Psi(\gamma_s \Delta t),$$
with $\gamma_i \in [-1, 1]$.
- Susuki scheme : $s = 5$, $p = 4$. Kahan-Li scheme: $s = 9$, $p = 6$.

New second-order scheme

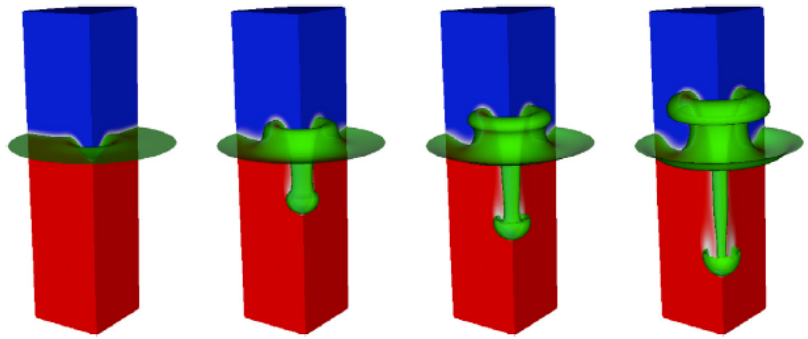
- The current second-order scheme is:
$$\Psi(\Delta t) = T\left(\frac{\Delta t}{2}\right) \circ R(\Delta t, \omega = 2) \circ T\left(\frac{\Delta t}{2}\right).$$
- It satisfies the time symmetry, but not $\Psi(0) = I_d$ for $\varepsilon \approx 0$. Indeed,
$$R(\Delta t = 0, \omega = 2) \iff \boldsymbol{f}^n = 2\boldsymbol{f}^{eq} - \boldsymbol{f}^n \neq \boldsymbol{f}^n$$
- However, $R(0, \omega = 2) \circ R(0, \omega = 2) = I_d$, and so we propose the following second-order scheme:
$$\Psi_{ap}(\Delta t) = T\left(\frac{\Delta t}{4}\right) \circ R(\Delta t, \omega = 2) \circ T\left(\frac{\Delta t}{2}\right) \circ R(\Delta t, \omega = 2) \circ T\left(\frac{\Delta t}{4}\right)$$

Implicit scheme based on kinetic relaxation III

- Error lines for the isothermal Euler equations. We have taken a CFL condition equal to 5 times the explicit one.



- Rayleigh-Taylor instability



- Theory and parallelization: Coulette and al [17-18-19].

Implicit scheme based on kinetic relaxation IV

- We have applied this strategy to the Guiding center for plasma physics, Helie [22]:

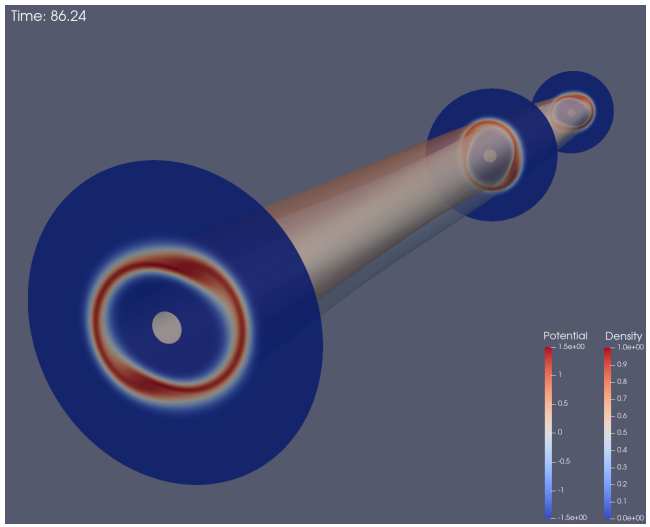
$$\begin{cases} \partial_t \rho - \nabla \cdot \left(((\nabla \phi)^{\perp} + \mathbf{B}) \rho \right) = 0, \\ -\Delta \phi = \rho - \int \rho dx, \end{cases}$$

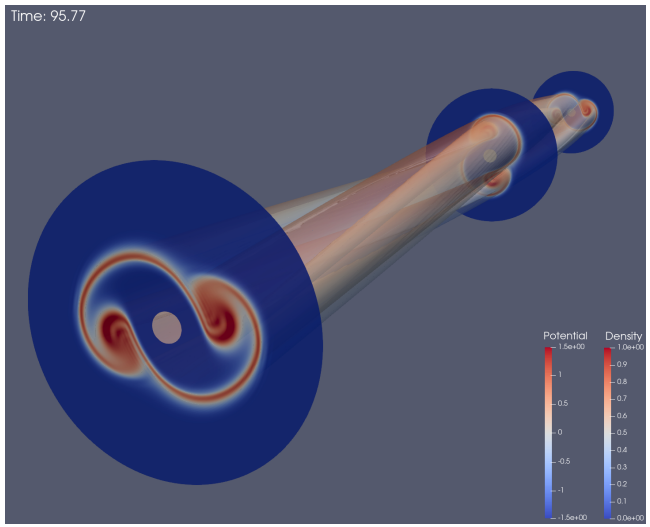
with $\mathbf{B} = (-b_\theta \sin(\theta), b_\theta \sin(\theta), b_\phi)^t$. We choose $b_\theta = 0.1$ and $b_\phi = 200$.

- **Scheme**: Exact transport in the toroidal direction, implicit DG kinetic scheme in the poloidal plane.
- **CFL conditions** for the classical and new schemes:

$$\Delta t_{exp} < \frac{\min(v_{pol}, v_\phi)}{\max(\Delta x_{pol}, \Delta x_\phi)} \qquad \rightarrow \qquad \Delta t_{new} < \frac{v_\phi}{\Delta x_\phi}$$

- **Test case**: 3D Diocotron instability. CFL condition equal to 33 times the explicit one.





Drawback: low Mach regime

- Kinetic relaxation: there exists specific choices of $\mathbf{f}^{eq}$ accurate in the isothermal low Mach regime (LBM community). **Drawback**: instabilities in other regimes.

Numerical error

- Error for the first order splitting scheme for Xin-Jin and vectorial kinetic relaxation:

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F}(\boldsymbol{U}) = \Delta t \left(\frac{2-\omega}{\omega}\right) \partial_x((\lambda^2 I_d - |A(\boldsymbol{U})|^2)\partial_x \boldsymbol{U}) + O(\Delta t^2)$$

- In the low Mach regime, $\partial_x u \approx M$, $\partial_x p \approx M$ and $c \approx \frac{1}{M}$, and so

$$\partial_t \rho + \partial_x(\rho u) \approx \Delta t \left(\frac{2-\omega}{\omega}\right) u^2 \left(\partial_x \left(\frac{1}{M^2} - 1\right) \partial_x \rho\right) + O(\Delta t^2)$$

- **Conclusion**: Too much diffusion on the contact wave.
- In the 2D case:

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p \approx \left(\frac{2-\omega}{\omega}\right) \frac{\Delta t}{2M^2} |\mathbf{u}|^2 \Delta \mathbf{u} + O(\Delta t^2)$$

- **Conclusion**: Too much diffusion on the shear wave.

- In Courtès and al [21], we proposed a kinetic relaxation valid for the low Mach regime in 1D, although proving its stability was not an easy task.

Suliciu relaxation for the Low-Mach regime I

- **Idea**: Linearize only the fast wave with relaxation.
- Non-conservative form and acoustic terms:

$$\left\{ \begin{array}{l} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t p + u\partial_x p + \rho c^2 \partial_x u = 0 \\ \partial_t u + u\partial_x u + \frac{1}{\rho}\partial_x p = 0 \end{array} \right.$$

Suliciu relaxation for the Low-Mach regime I

- **Idea**: Linearize only the fast wave with relaxation.
- Non-conservative form and acoustic terms:

$$\left\{\begin{array}{l} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t p + u\partial_x p + \rho c^2 \partial_x u = 0 \\ \partial_t u + u\partial_x u + \frac{1}{\rho}\partial_x p = 0 \end{array}\right.$$

- **Idea**: Relax only the acoustic part ([BCG18]) to linearize the implicit part.

$$\left\{\begin{array}{l} \partial_t \rho + \partial_x(\rho v) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u v + \Pi) = 0 \\ \partial_t E + \partial_x(E v + \Pi v) = 0 \\ \partial_t \Pi + v\partial_x \Pi + \phi\lambda^2 \partial_x v = \frac{1}{\varepsilon}(p - \Pi) \\ \partial_t v + v\partial_x v + \frac{1}{\phi}\partial_x \Pi = \frac{1}{\varepsilon}(u - v) \end{array}\right.$$

Suliciu relaxation for the Low-Mach regime I

- **Idea**: Linearize only the fast wave with relaxation.
- Non-conservative form and acoustic terms:

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t p + u\partial_x p + \rho c^2 \partial_x u = 0 \\ \partial_t u + u\partial_x u + \frac{1}{\rho}\partial_x p = 0 \end{cases}$$

- **Idea**: Relax only the acoustic part ([BCG18]) to linearize the implicit part.

$$\begin{cases} \partial_t \rho + \partial_x(\rho v) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u v + \Pi) = 0 \\ \partial_t E + \partial_x(E v + \Pi v) = 0 \\ \partial_t \Pi + v\partial_x \Pi + \phi\lambda^2 \partial_x v = \frac{1}{\varepsilon}(p - \Pi) \\ \partial_t v + v\partial_x v + \frac{1}{\phi}\partial_x \Pi = \frac{1}{\varepsilon}(u - v) \end{cases}$$

- **Limit**:

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = \varepsilon \partial_x \left[A \partial_x p \right] \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = \varepsilon \partial_x \left[(A u \partial_x p) + B \partial_x u \right] \\ \partial_t E + \partial_x(E u + p u) = \varepsilon \partial_x \left[AE \partial_x p + A \partial_x \frac{p^2}{2} + B \partial_x \frac{u^2}{2} \right], \end{cases}$$

  with $A = \frac{1}{\rho}\left(\frac{\rho}{\phi} - 1\right)$ and $B = \left(\rho\phi\lambda^2 - \rho^2 c^2\right)$.
- **Stability**: $\phi\lambda > \rho c^2$ and $\rho > \phi$.

Suliciu relaxation for the Low-Mach regime I

- **Idea**: Linearize only the fast wave with relaxation.
- Non-conservative form and acoustic terms:

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = 0 \\ \partial_t p + u\partial_x p + \rho c^2 \partial_x u = 0 \\ \partial_t u + u\partial_x u + \frac{1}{\rho}\partial_x p = 0 \end{cases}$$

- **Idea**: Relax only the acoustic part ([BCG18]) to linearize the implicit part.

$$\begin{cases} \partial_t \rho + \partial_x(\rho v) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u v + \Pi) = 0 \\ \partial_t E + \partial_x(E v + \Pi v) = 0 \\ \partial_t \Pi + v\partial_x \Pi + \phi\lambda^2 \partial_x v = \frac{1}{\varepsilon}(p - \Pi) \\ \partial_t v + v\partial_x v + \frac{1}{\phi}\partial_x \Pi = \frac{1}{\varepsilon}(u - v) \end{cases}$$

- **Limit**:

$$\begin{cases} \partial_t \rho + \partial_x(\rho u) = \varepsilon \partial_x \left[A \partial_x p \right] \\ \partial_t(\rho u) + \partial_x(\rho u^2 + p) = \varepsilon \partial_x \left[(A u \partial_x p) + B \partial_x u \right] \\ \partial_t E + \partial_x(E u + p u) = \varepsilon \partial_x \left[AE \partial_x p + A \partial_x \frac{p^2}{2} + B \partial_x \frac{u^2}{2} \right], \end{cases}$$

  with $A = \frac{1}{\rho}\left(\frac{\rho}{\phi} - 1\right)$ and $B = \left(\rho\phi\lambda^2 - \rho^2 c^2\right)$.
- **Stability**: $\phi\lambda > \rho c^2$ and $\rho > \phi$.

Avdantage

We keep the conservative form for the original variables and obtain fully linear acoustics.

Suliciu relaxation for Low-Mach II

➜ Splitting: Convective part treated explicitly / Acoustic part treated implicitly.

$$\begin{cases} \partial_t \rho + \partial_x(\rho v) = 0 \\ \partial_t(\rho u) + \partial_x(\rho uv + \mathcal{M}^2(t)\Pi) = 0 \\ \partial_t E + \partial_x(Ev + \mathcal{M}^2(t)\Pi v) = 0 \\ \partial_t \Pi + v\partial_x \Pi + \phi\lambda_c^2 \partial_x v = 0 \\ \partial_t v + v\partial_x v + \dfrac{\mathcal{M}^2(t)}{\phi}\partial_x \Pi = 0 \end{cases} \quad \text{and} \quad \begin{cases} \partial_t \rho = 0 \\ \partial_t(\rho u) + (1-\mathcal{M}^2(t))\partial_x \Pi = 0 \\ \partial_t E + (1-\mathcal{M}^2(t))\partial_x(\Pi v) = 0 \\ \partial_t \Pi + \phi(1-\mathcal{M}^2(t))\lambda_a^2 \partial_x v = 0 \\ \partial_t v + (1-\mathcal{M}^2(t))\dfrac{1}{\phi}\partial_x \Pi = 0 \end{cases}$$

with $\mathcal{M}(t) \approx \max\left(\mathcal{M}_{min}, \min\left(\max \dfrac{|u|}{c}, 1\right)\right)$.

Suliciu relaxation for Low-Mach II

- Splitting: Convective part treated explicitly / Acoustic part treated implicitly.

$$\begin{cases} \partial_t \rho + \partial_x(\rho v) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u v + \mathcal{M}^2(t)\Pi) = 0 \\ \partial_t E + \partial_x(Ev + \mathcal{M}^2(t)\Pi v) = 0 \\ \partial_t \Pi + v\partial_x \Pi + \phi\lambda_c^2 \partial_x v = 0 \\ \partial_t v + v\partial_x v + \dfrac{\mathcal{M}^2(t)}{\phi}\partial_x \Pi = 0 \end{cases} \quad \text{and} \quad \begin{cases} \partial_t \rho = 0 \\ \partial_t(\rho u) + (1-\mathcal{M}^2(t))\partial_x \Pi = 0 \\ \partial_t E + (1-\mathcal{M}^2(t))\partial_x(\Pi v) = 0 \\ \partial_t \Pi + \phi(1-\mathcal{M}^2(t))\lambda_a^2 \partial_x v = 0 \\ \partial_t v + (1-\mathcal{M}^2(t))\dfrac{1}{\phi}\partial_x \Pi = 0 \end{cases}$$

with $\mathcal{M}(t) \approx \max\left(\mathcal{M}_{min}, \min\left(\max \dfrac{|u|}{c}, 1\right)\right)$.

- Eigenvalues: explicit part: v, $v \pm \mathcal{M}(t) \underbrace{\lambda_c}_{\approx c}$; implicit part: 0, $\pm(1-\mathcal{M}^2(t)) \underbrace{\lambda_a}_{\approx c}$.
  - Step 1: we solve

$$(I_d - (1-\mathcal{M}^2(t_n))^2 \Delta t^2 \lambda_a^2 \partial_{xx})\Pi^{n+1} = \Pi^n - \Delta t(1-\mathcal{M}^2(t_n))\phi\lambda_a^2 \partial_x v^n$$

  - Step 2: we compute v^{n+1} and ρu^{n+1} using Π^{n+1}, and E^{v+1} using $\Pi^{n+1} v^{n+1}$.

Suliciu relaxation for Low-Mach II

- Splitting: Convective part treated explicitly / Acoustic part treated implicitly.

$$\begin{cases} \partial_t \rho + \partial_x(\rho v) = 0 \\ \partial_t(\rho u) + \partial_x(\rho u v + \mathcal{M}^2(t)\Pi) = 0 \\ \partial_t E + \partial_x(Ev + \mathcal{M}^2(t)\Pi v) = 0 \\ \partial_t \Pi + v\partial_x \Pi + \phi\lambda_c^2 \partial_x v = 0 \\ \partial_t v + v\partial_x v + \dfrac{\mathcal{M}^2(t)}{\phi}\partial_x \Pi = 0 \end{cases} \quad \text{and} \quad \begin{cases} \partial_t \rho = 0 \\ \partial_t(\rho u) + (1-\mathcal{M}^2(t))\partial_x \Pi = 0 \\ \partial_t E + (1-\mathcal{M}^2(t))\partial_x(\Pi v) = 0 \\ \partial_t \Pi + \phi(1-\mathcal{M}^2(t))\lambda_a^2 \partial_x v = 0 \\ \partial_t v + (1-\mathcal{M}^2(t))\dfrac{1}{\phi}\partial_x \Pi = 0 \end{cases}$$

with $\mathcal{M}(t) \approx \max\left(\mathcal{M}_{min}, \min\left(\max \dfrac{|u|}{c}, 1\right)\right)$.

- Eigenvalues: explicit part: v, $v \pm \mathcal{M}(t) \underbrace{\lambda_c}_{\approx c}$; implicit part: 0, $\pm(1-\mathcal{M}^2(t)) \underbrace{\lambda_a}_{\approx c}$.
  - Step 1: we solve

$$(I_d - (1-\mathcal{M}^2(t_n))^2 \Delta t^2 \lambda_a^2 \partial_{xx})\Pi^{n+1} = \Pi^n - \Delta t(1-\mathcal{M}^2(t_n))\phi\lambda_a^2 \partial_x v^n$$

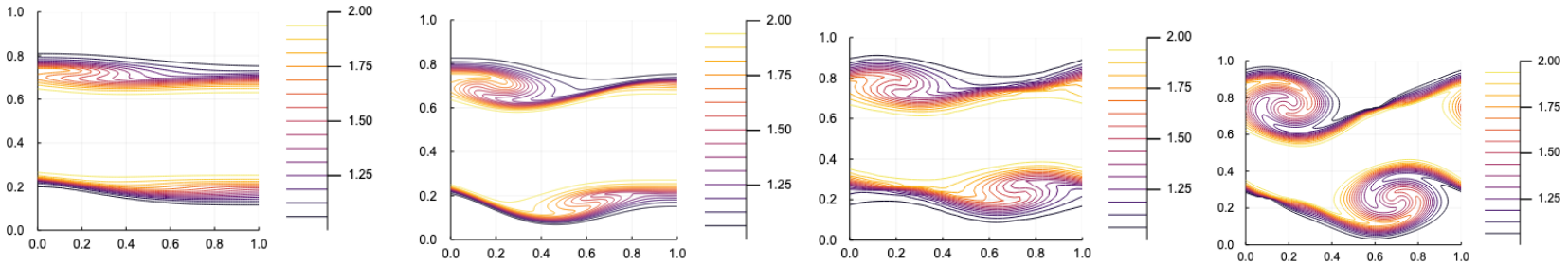
  - Step 2: we compute v^{n+1} and ρu^{n+1} using Π^{n+1}, and E^{v+1} using $\Pi^{n+1} v^{n+1}$.

Advantages

- We construct an efficient Godunov relaxation scheme for the explicit part.
- We solve only a linear and constant Laplacian. The matrix is only assembled once.
- No conditioning issues coming from large gradients of ρ.

Results in 2D: Kelvin-Helmholtz instability

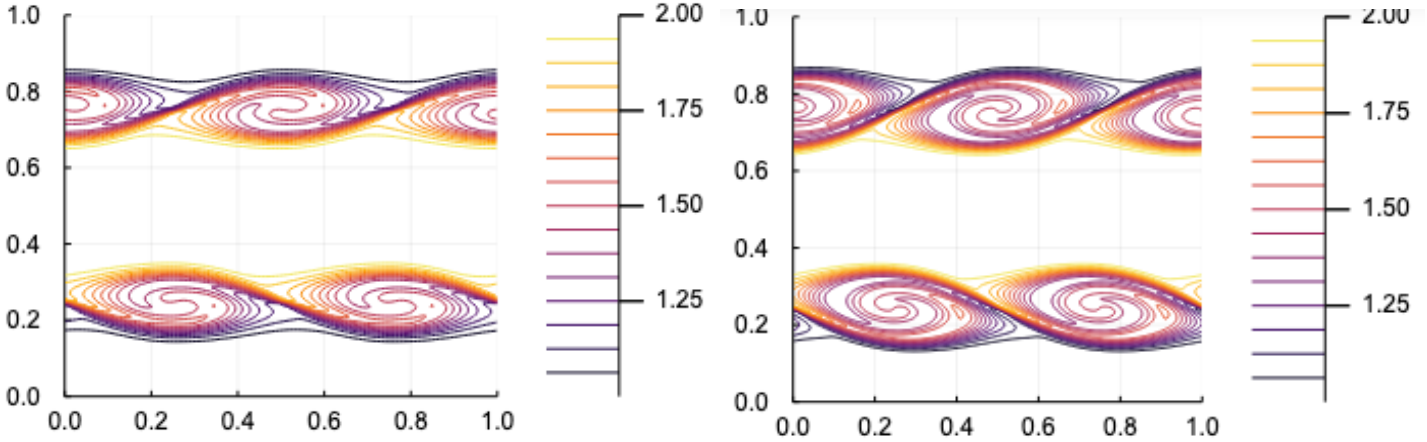
- Kelvin-Helmholtz instability. Density:



- Density at time $T_f = 3$, $k = 1$, $M_0 = 0.1$. Explicit Lagrange-Remap scheme with 120×120 (left) and 360×360 cells (middle left), SI two-speed relaxation scheme ($\lambda_c = 18$, $\lambda_a = 15$, $\phi = 0.98$) with 42×42 (middle right) and 120×120 cells (right).

Results in 2D: Kelvin-Helmholtz instability

- Kelvin-Helmholtz instability. Density:



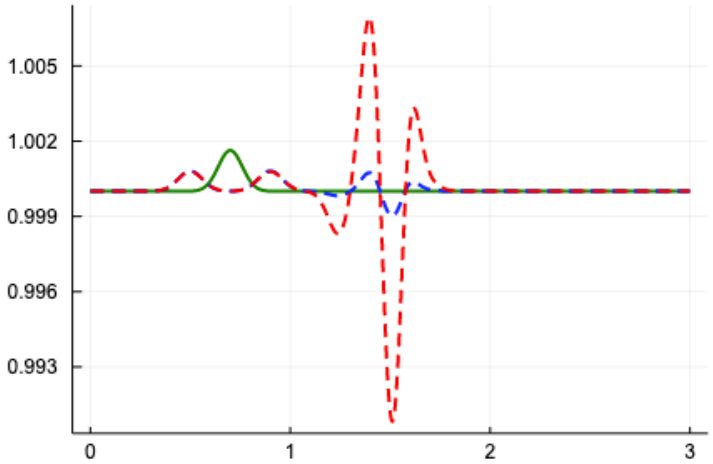
- Density at time $T_f = 3$, $k = 2$, $M_0 = 0.01$ with SI two-speed relaxation scheme ($\lambda_c = 180$, $\lambda_a = 150$, $\phi = 0.98$). Left: 120×120 cells. Right: 240×240 cells.

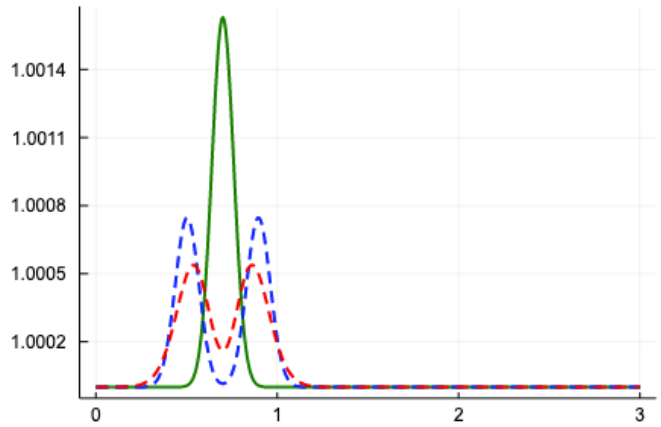
Well balanced extension

Idea

For the Ripa Shallow Water model, we coupled preivous semi-implicit with WB method (Jin-Levermore [98]). We obtain a WB scheme with time step independant from the "Mach" number.

➜ Test case: equilibrium perturbation:





➜ We use that discrete steady state are exactly consistant with continuous ones.

➜ **More general case:** Euler equations with gravity, equilibrium: $\boldsymbol{u} = 0, \quad \nabla \boldsymbol{p} = -\rho \nabla \phi$

Result

In Fra [16], we proposed a WB scheme for discrete steady states constructed with a high-order polynomial reconstruction. The scheme converges with order one and order q around the steady state.

Numerical solvers based on deep learning

Principle of PINNs

Deep learning

Deep learning has demonstrated its ability to build efficient high-dimensional models and functions. But can we use it to approximately solve PDEs ?

- First method: Physics-Informed neural networks (PINNs) (Raissi & Karniadakis, 2017).
- Consider the following PDE, with α, β some parameters:

$$\begin{cases} \partial_t \mathbf{U} = \mathcal{N}(\mathbf{U}, \partial_x \mathbf{U}, \partial_{xx} \mathbf{U}, \beta) \\ \mathbf{U}_h(t,x) = g(x), \quad \forall x \in \partial\Omega \\ \mathbf{U}(0,x) = \mathbf{U}_0(x, \alpha). \end{cases}$$

- **Idea I**: represent/approximate solutions of PDEs by a Neural Network $\mathbf{U}_\theta(t,x)$.
- **Idea II**: Since neural networks can be $C^p(\mathbb{R}^d)$, we can exactly compute the PDE residual.

PINNs recast as an optimization problem – exact integration

$$\min_\theta J_r(\theta) + J_b(\theta) + J_i(\theta)$$

with

$$J_r(\theta) = \int_0^T \int_\Omega \| \partial_t \mathbf{U}_\theta(t,x) - \mathcal{L}(\mathbf{U}_\theta, \partial_x \mathbf{U}_\theta, \partial_{xx} \mathbf{U}_\theta, \beta)(t,x) \|_2^2 \, dxdt$$

$$J_b(\theta) = \int_0^T \int_{\partial\Omega} \| \mathbf{U}_\theta(t,x) - g(x) \|_2^2 \, dxdt, \quad J_i(\theta) = \int_\Omega \| \mathbf{U}_\theta(0,x) - \mathbf{U}_0(x) \|_2^2$$

Principle of PINNs

Deep learning

Deep learning has demonstrated its ability to build efficient high-dimensional models and functions. But can we use it to approximately solve PDEs ?

- First method: Physics-Informed neural networks (PINNs) (Raissi & Karniadakis, 2017).
- Consider the following PDE, with α, β some parameters:

$$\begin{cases} \partial_t \mathbf{U} = \mathcal{N}(\mathbf{U}, \partial_x \mathbf{U}, \partial_{xx} \mathbf{U}, \beta) \\ \mathbf{U}_h(t,x) = g(x), \quad \forall x \in \partial\Omega \\ \mathbf{U}(0,x) = \mathbf{U}_0(x, \alpha). \end{cases}$$

- **Idea I**: represent/approximate solutions of PDEs by a Neural Network $\mathbf{U}_\theta(t,x)$.
- **Idea II**: Since neural networks can be $C^p(\mathbb{R}^d)$, we can exactly compute the PDE residual.

PINNs recast as an optimization problem – Monte-Carlo integration

$$\min_\theta J_r(\theta) + J_b(\theta) + J_i(\theta)$$

with

$$J_r(\theta) = \sum_{i=1}^{N} \| \partial_t \mathbf{U}_\theta(t_i,x_i) - \mathcal{L}(\mathbf{U}_\theta, \partial_x \mathbf{U}_\theta, \partial_{xx} \mathbf{U}_\theta, \mu)(t_i,x_i) \|_2^2$$

$$J_b(\theta) = \sum_{i=1}^{N_b} \| \mathbf{U}_\theta(t_i,x_i) - g(x_i) \|_2^2, \quad J_i(\theta) = \sum_{i=1}^{N_i} \| \mathbf{U}_\theta(0,x_i) - \mathbf{U}_0(x_i, \alpha) \|_2^2$$

Parametric PINNs

- **Advantages of PINNs**: mesh-less approach, not too sensitive to the dimension.
- **Main drawback of PINNs**: they are not competitive with classical methods.
- Training for the 1D viscous Burgers equation with $\nu = 10^{-4}$: 2 hours
- Interesting possibilities: use the strengths of PINNs to solve parametric PDEs.

Parametric PINNs

- The neural network becomes $\mathbf{U}_\theta(t, x, \alpha, \beta)$.
- We also sample the parameter space.

- Training for viscous Burgers 1D equation with $\nu = [10^{-2}, 10^{-4}]$:$\approx$ 2 hours

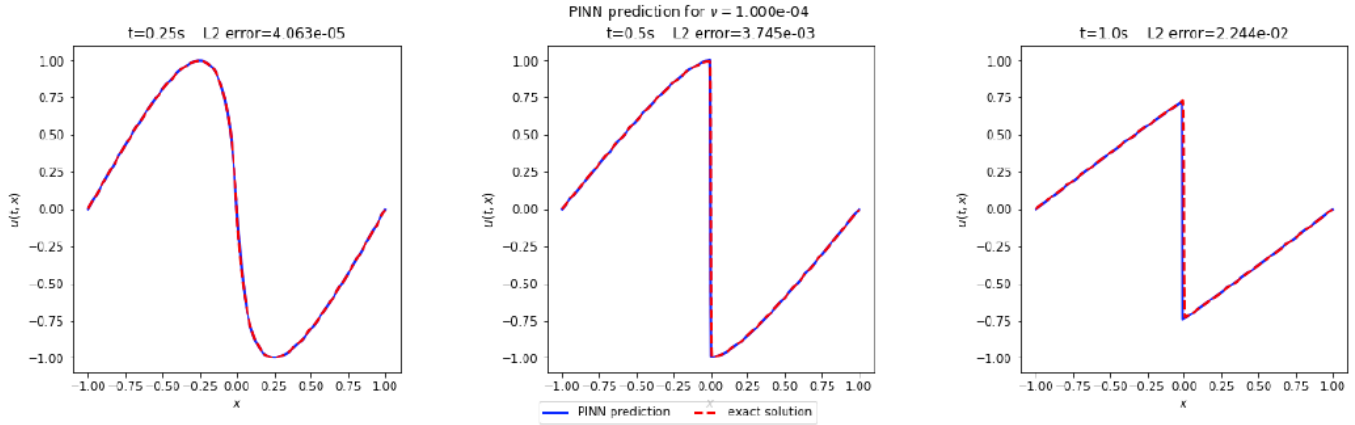


Figure 5. Burgers' equation: Comparison of the predicted and exact solutions corresponding to three temporal snapshots. $\nu = 10^{-4}$

Neural operators

- How to go beyond parametric solutions ?
- We consider the following toy problem:

$$\begin{cases} -\nabla \cdot (a(\boldsymbol{x})\nabla u) = f(\boldsymbol{x}), & \forall x \in \Omega \\ u = g, & \forall x \in \partial\Omega \end{cases}$$

- Formally, there exists a pseudo inverse operator G^+, such that $G^+(f(\boldsymbol{x}), g, a(\boldsymbol{x})) = u(\boldsymbol{x})$.

Operator learning

Approximate G^+ by a neural network on a subspace of the data.

- CNNs and **neural operators** can be used to approximate G^+.
- The solution is given by

$$u(x) = \int_\Omega G_a(x, y) f(y) dy$$

with G_a a Green kernel. **Important**: the operator is nonlocal.

Neural operator layer

The layer transforms a function $v_l(x)$ into a function $v_{l+1}(x)$, which has the form:

$$\forall x \in D,\ v_{l+1}(x) = \sigma_{l+1}\left(W_l v_l(x) + b(x) + \int_D k(x, y) v_l(y) d\nu(y),\right)$$

where $W_t \in \mathbb{R}^{d,d}$ is a weight matrix, and $k(x, y) \in C^p(D \times D; \mathbb{R}^{n_{t+1}} \times \mathbb{R}^{n_t})$.

Neural operators: an example

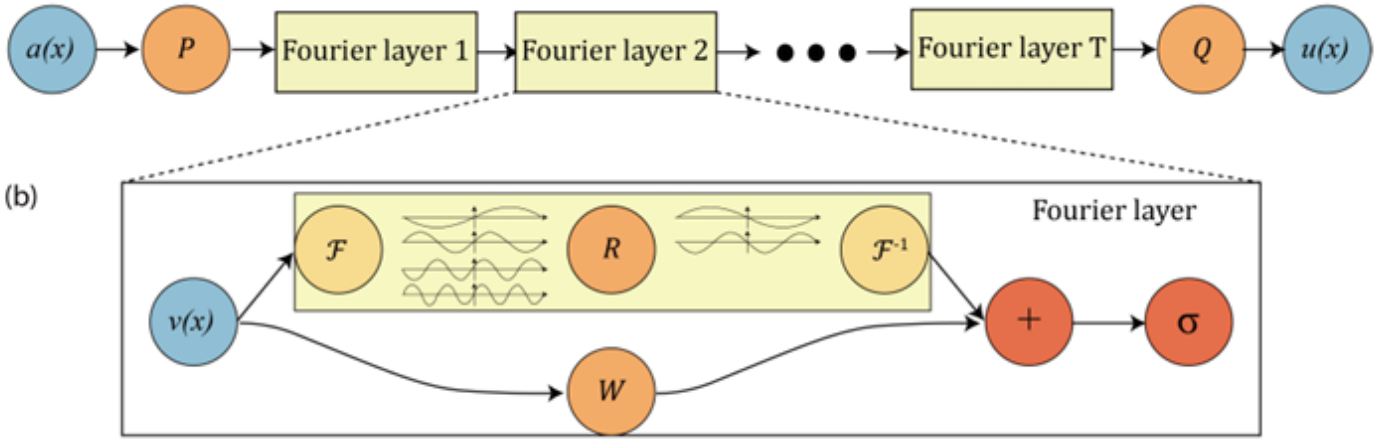
Fourier Neural Network

The neural operator layer of FNOs is an integral kernel (Zi and al [20]):

$$\int_D k(x,y)v(y)d\nu(y) \approx \mathcal{F}^{-1}(R_\theta \mathcal{F}(v(x)),$$

with R_θ the learnable filters in the Fourier space.

Principle:



Main objective

Remark

For a low variability of parameters, neural operators and parametric PINNs are able to quickly predict a solution. For a larger variability of the parameters, these approaches seem to lose accuracy.

Question

How to accelerate a numerical solver using neural networks, while retaining the accuracy and stability of the numerical methods?

Aim

Incorporate the neural networks into numerical methods to increase the efficiency while keeping the properties of the method.

First idea: Newton's method with data-driven initialization

We wish to solve the following elliptic problem:

$$u - \alpha_0 \nabla \cdot (A(x,y)k(u)\nabla u) = f(x,y)$$

Solver

- Discretization with finite differences or finite elements (on structured meshes for the moment)
- Newton-Krylov method (Jacobian-free approximation + GMRES for linear part)
- After discretization, we solve the problem: $G_{A_h,f_h,\alpha_0}(\mathbf{u}_h) = 0$

Idea

Train a Fourier Neural Operator (FNO) to approximate the solution of the elliptic equation and use it as an initial guess.

Algorithm:

- Fix a mesh, fix α_0 and $k(\cdot)$,
- Randomly generate lots of data points u_h^i, A_h^i (sum of random Gaussian functions),
- Compute the right-hand side associated with f_h^i,
- Train the neural network $G_\theta^+(A_h^i, f_h^i)$, by minimizing

$$J(\theta) = \omega \sum_{i=1}^{n} \| G_\theta^+(A_h^i, f_h^i) - u_h^i \| + (1-\omega) \sum_{i=1}^{n} \| G_{A_h^i, f_h^i, \alpha_0}(G_\theta^+(A_h^i, f_h^i)) \|.$$

Newton's method with data-driven initialization: results

- We solve the 1D case with $k(u) = u^4$, $a(x)$ a random function.
- We compare the average results for different α_0 (the larger α_0 is, the more the problem is nonlinear):

mesh	$\alpha_0 = 2$ (40 sim)	$\alpha_0 = 5$ (25 sim)	$\alpha_0 = 8$ (25 sim)
100 cells	+500%	+1800%	+5000%
200 cells	+88%	+230%	+620%
400 cells	+82%	+150%	+220%
600 cells	+92%	+220%	+250%

Table: Comparison of the mean "gain" for different values of α_0.

- **Fails**: on all the tests, we have 0% of fail (our method being less efficient than the classical one) for the iteration criterion, and around 2% of fail on the CPU time criterion.
- On more refined meshes, the gain is smaller (the network acts only at the beginning of the convergence).
- The method is more efficient for highly nonlinear systems.

Second idea: NN-enhanced finite element solver

Question

With parametric PINNs or neural operators, we accelerated a solver (Newton's method). Can we also increase the accuracy of the method ?

- **Classical method for elliptic problem**: Finite elements (FE).
- Strong form $\rightarrow$ weak form:

$$-\Delta u = f \qquad \rightarrow \qquad \int_\Omega \nabla u \cdot \nabla v dx = \int_\Omega f v dx, \quad v \in H_0^1(\Omega)$$

- To discretize on a mesh with N cells, we use:

$$u_h = \sum_{j=0}^{N} \alpha_j \phi_j(x), \quad v = \phi_i$$

with $\phi_i(x)$ the hat function.

Idea

If we have an approximation of the solution given by a neural network (here parametric PINNs) $u_\theta(x, \boldsymbol{\mu})$ we can use to increase the representation of the solution

In the modified FE scheme, we use:

$$u_h = \sum_{j=0}^{N} \alpha_j \phi_j(x) u_\theta(x, \boldsymbol{\mu}), \quad v = \phi_i(x) u_\theta(x, \boldsymbol{\mu}).$$

NN-enhanced finite element solver: results

- First test:

$$-\partial_{xx} u = \alpha \sin(2\pi x) + \beta \sin(4\pi x) + \gamma \sin(8\pi x)$$

We train with $(a, b, c) \in [0, 1]^3$ and test with $(a, b, c) \in [0, 1.2]^3$.

Results with 20 cells:

Data set	inside training set	outside training set	inside/outside
Average gain vs FE	101	28.4	76.8
Average gain vs PINNS	6.7	7	6.8

- Second test:

$$v\partial_x u - \frac{v}{Pe}\partial_{xx} u = r$$

We train with $r \in [1.5, 2.0]$, $v \in [1.5, 2.0]$ and $Pe \in [10, 120]$,
and we test with $r \in [1.5, 2.2]$, $v \in [1.5, 2.2]$ and $Pe \in [10, 150]$.

Results with 20 cells:

Data set:	inside training set	outside training set	inside/outside
Average gain	110	25	81.6

- The method converges with second-order accuracy.
- **Next step, with Inria Pau**: 2D extension, theory of convergence, extension to time problems.

Research program

Research program I

My research program

My research program, around four axes, is fully in line with that of the new Inria team

First Axis: data-driven solver

- ➜ design hybrid solvers with ML and classical methods using supervised learning and optimal control approaches (differentiable physics or reinforcement learning).
- **Applications**: viscosity and LBM scheme (ongoing work with L. Bois PhD), slope limiting, flux correction, filter the numerical dispersion (PhD N. Victorion) etc.
- ➜ Enhanced Finite element and DG scheme in time (with Inria Makutu, Mimesis). Training using PINNs (as is done now), neural operators or differentiable physics approaches.
- ➜ Acceleration of implicit codes / Newton /inverse problems using neural operators (with Inria Atlantis and Makutu in the PEPR NumpEx) or super-resolution (with Inria Mimesis). Specific focus on unstructured meshes, multi-scale and nonlinear PDEs.

Second Axis: Reduced models

- ➜ Design structure-preserving reduced models with ML for collisional and oscillatory Vlasov equations. **Type of models**: Hamiltonian nonlinear model order reduction and hyperbolic moment approximations (ANR MILK with Munich, PhD of G. Steimer, postdoc of Y. Nasseri and L. Tremant).

Research program II

Second Axis: Reduced models

- Reduced basis based on PINNs (future work with Inria Atlantis and Makutu in the PEPR NumpEx).
- reduced-order modeling and hyper-reduction for hyperbolic systems using control optimal approaches like differential physics, RL, PINNs (PhD of M. Bestard).

Third Axis: Optimal control

- Acceleration of optimal control using reduced models (work of K. Lutz).
- Reinforcement learning in large dimensional control space. How to explore the control space? Application: quantum control, inverse problems and scheme optimization.

Last axis: self-specializing code

- It is difficult to pre-train neural networks used in numerical methods on large set of parameters (initial data, coefficients, etc).
- **Assumption**: An instance of code is often used in small parameter areas.
- Train the networks as the simulations progress and move towards codes that specialize and improve over time.

Thanks

Thanks for your attention!

Thanks to my colleagues and co-workers

Thanks to my friends and family

Thanks to my wife

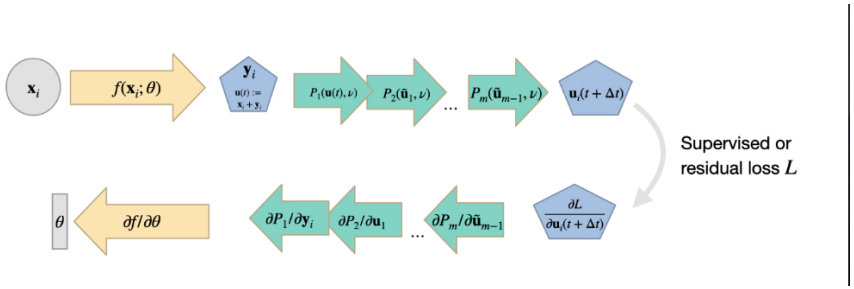
Differentiable physics

Principle

Differentiable physicss

Write the scheme such that we can apply automatic differentiation and back-propagation to compute the gradient of each function of the scheme in the code, and each composition of these functions.

- Using that, we can compute the gradient with respect to all inputs of the solver, or of sub-parts of the solver.
- **Consequences**: We can put a NN anywhere in the solver and optimize it with respect to a criterion on the simulation result.



Link with optimal control

In optimal control we compute the gradient of the loss with respect to the input with an adjoint method. It is another use of automatic differentiation methods.

Drawback

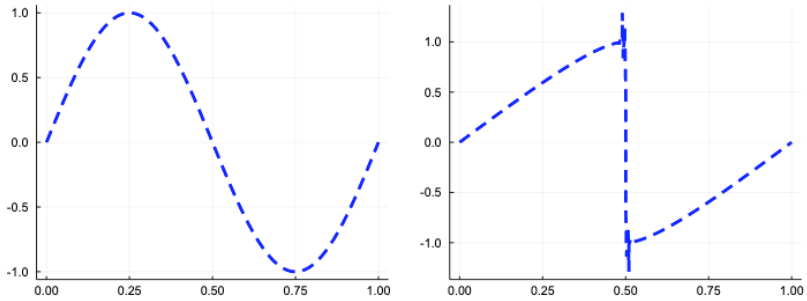
With back-propagation, stability problems (vanishing or exploding gradient) can arise when composing too many functions.

General problem

- We want to solve general hyperbolic PDEs:

$$\partial_t \mathbf{U} + \partial_x \mathbf{F}(\mathbf{U}) = 0$$

- High order method (MUSCL, HO finite volumes or DG) generate oscillations around areas with strong gradients or shock waves: Gibbs phenomenon.
- Example on the Burgers equation:



- **Solutions**: slope limiting, artificial viscosity, filtering, etc.

Goal

Design slope limiting for MUSCL or artificial viscosity for DG using neural networks.

- We have a DG scheme, written under the form

$$\partial_t^{rk} \mathbf{U}_h + \partial_x^{DG} \mathbf{F}(\mathbf{U}_h) = 0.$$

- Artificial viscosity method: add a diffusion operator, which acts on the oscillations.
- Modified scheme:

$$\partial_t^{rk} \mathbf{U}_h + \partial_x^{DG} \mathbf{F}(\mathbf{U}_h) = \partial_x^{DG} (\mathbf{D}(\mathbf{U}_h) \partial_x^{DG} \mathbb{U}_h).$$

Differentiable physicss approach I

Tool

We propose to use differentiable physicss (control optimal approach) to design new types of viscosity model.

- We define a NN $D_\theta(\mathbf{U}_h(t))$ with $\mathbf{U}_h(t)$ the discrete solution.

Goal

Our objective to find a solution of the minimization problem:

$$\min_\theta \int_{U_0} V_\theta(\mathbf{U}_0) dp(\mathbf{U}_0) d\mathbf{U}_0, \quad V_\theta^T(\mathbf{U}_0) = \int_0^T C(\mathbf{U}_h(t)) dt,$$

with $p(\mathbf{U}_0)$ a probability law of initial data on $\mathbf{U}_0$, with C a cost function and $\mathbf{U}_0 = \mathbf{U}_h(0)$ an initial condition.

- The transition between two time steps is given by $\mathbf{U}_h^{n+1} = S_h(\mathbf{U}_h^n, D_\theta(\mathbf{U}_h^n))$ with our scheme. As a consequence, after time discretization we have:

$$V_\theta^T(\mathbf{U}_0) = C(\mathbf{U}_0) + C(S_h(\mathbf{U}_0, D_\theta(\mathbf{U}_0))) + C(S_h(S_h(\mathbf{U}_0, D_\theta(\mathbf{U}_0)), D_\theta(S_h(\mathbf{U}_0, D_\theta(\mathbf{U}_0))))) + \dots,$$

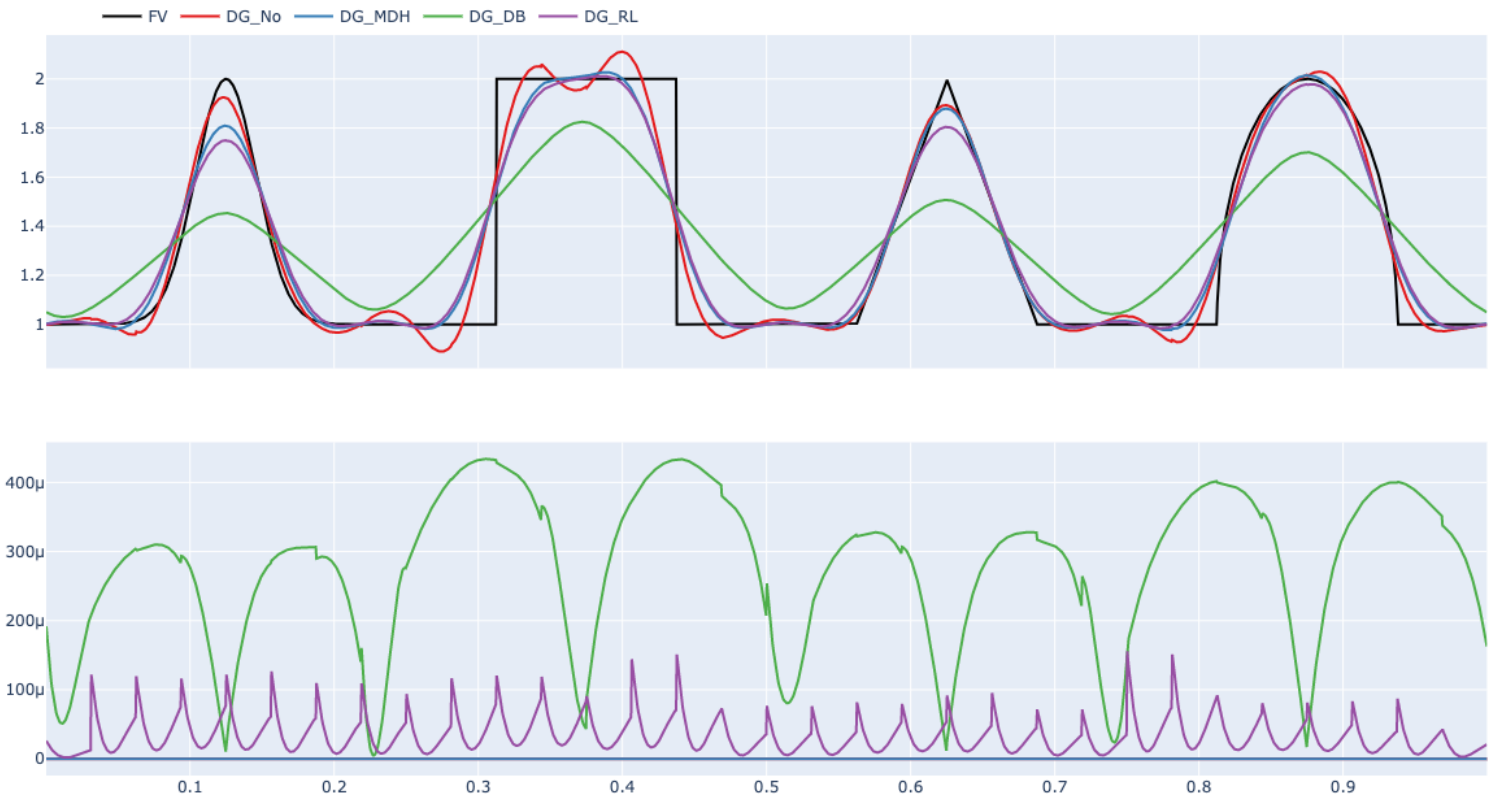
- As previously mentioned in we can compute by automatic differentiation:

$$\nabla_\theta V_\theta^T(\mathbf{U}_0)$$

which allows to solve the minimization problem on $J(\theta)$ using a gradient method.

Results I

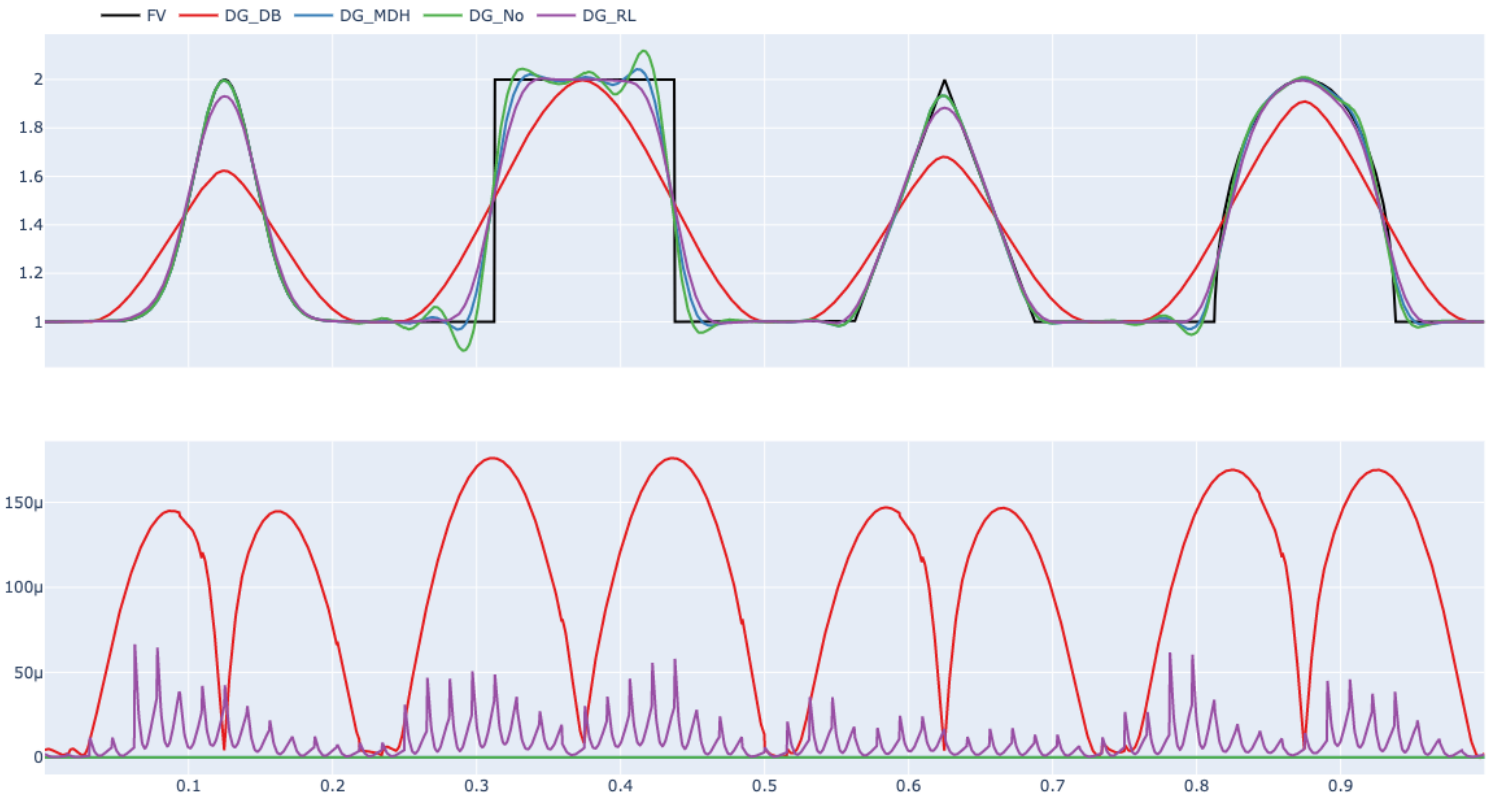
- We consider two loss: C_{error} and C_{osc}. We consider the grids: 32 cells, 64 cells and 128 cells.



- 32 cells

Results I

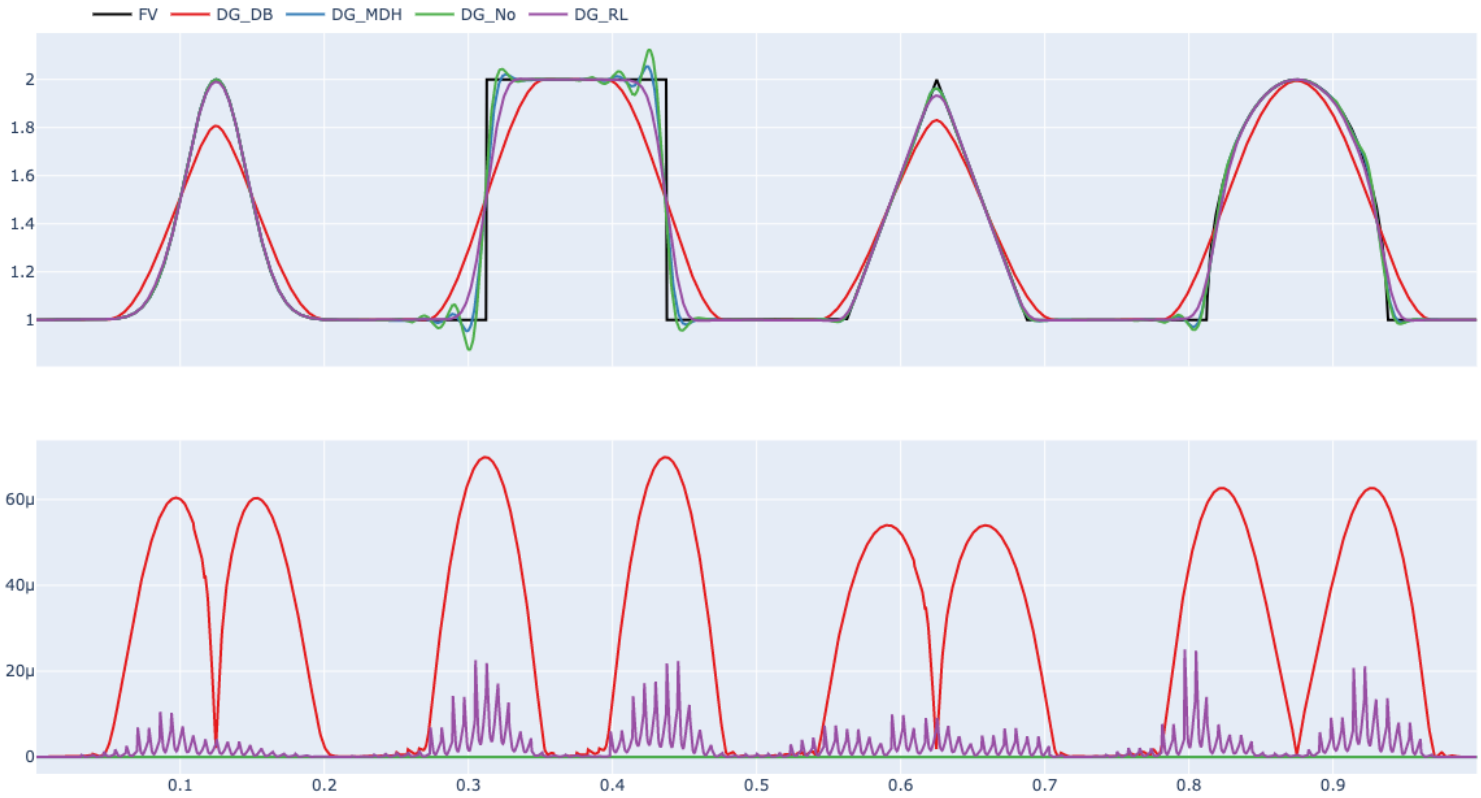
- We consider two loss: C_{error} and C_{osc}. We consider the grids: 32 cells, 64 cells and 128 cells.



- 64 cells

Results I

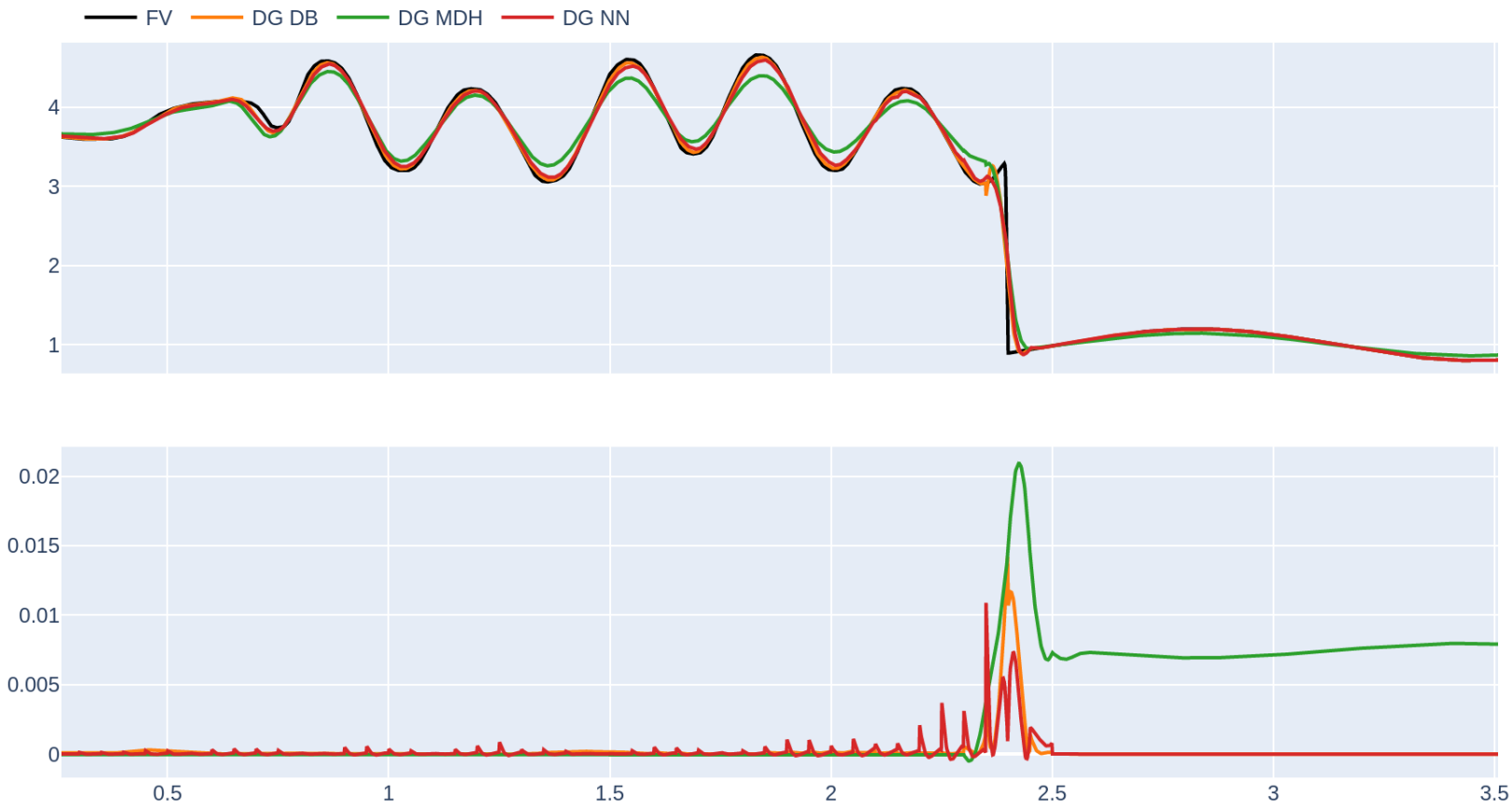
- We consider two loss: C_{error} and C_{osc}. We consider the grids: 32 cells, 64 cells and 128 cells.



- 128 cells

Results II

- Euler equations, viscosity model: $D_\theta(\rho, U, E)$ on the three equations.
- Shu-Osher problem:



- 200 cells